CSE 222

HW 05

MUSTAFA AKILLI

131044017

Projenin Github Linki:

https://github.com/AKILLIMUSTAFA/CSE222-HW05

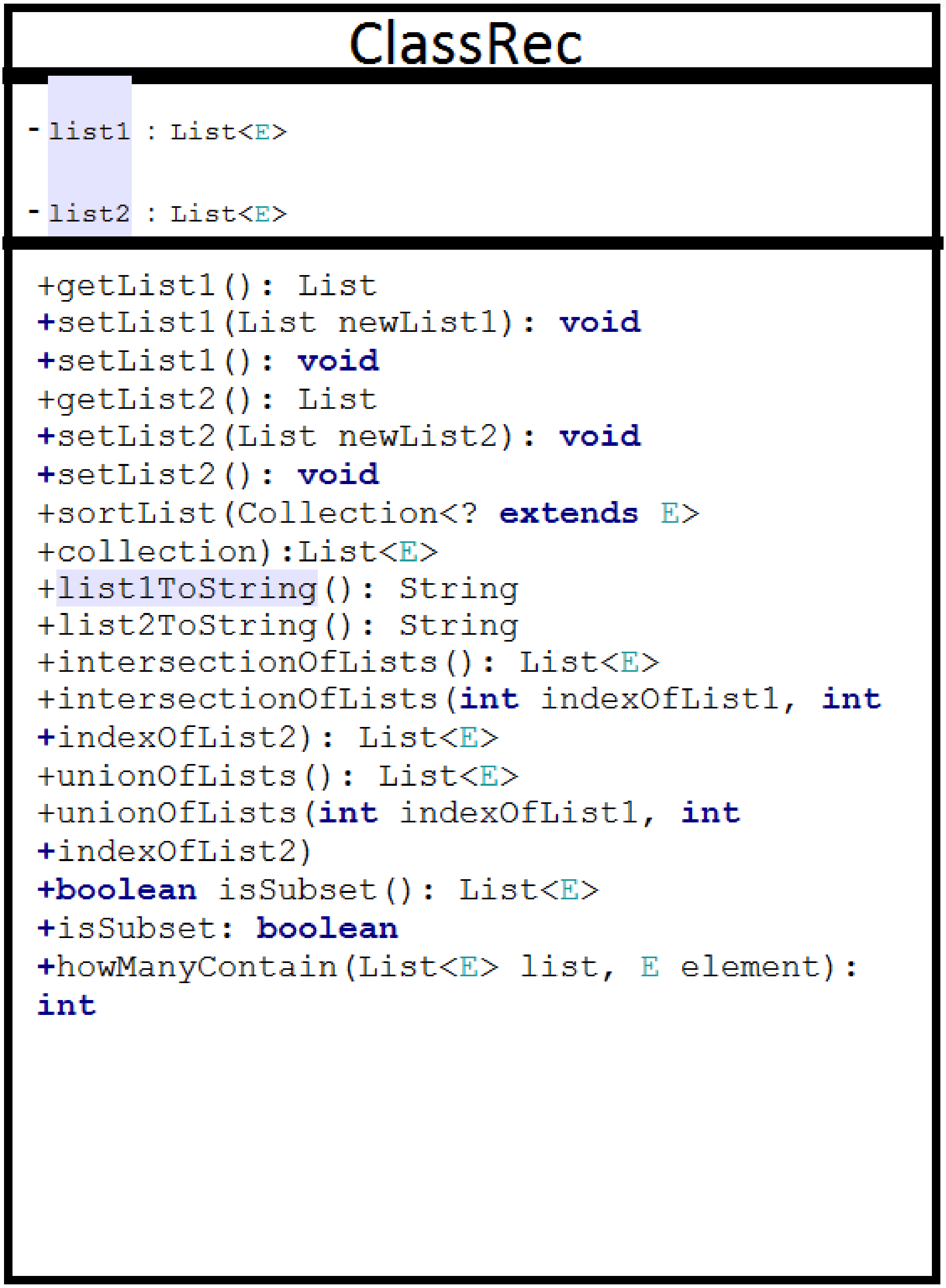
Detailed system requirements

Bu ödevde 3 part bulunmaktadır. İlk partta Towers of Hanoi probleminin iterative olarak çözümü istenmekte. Part 2 de LinkListRec classı içine yeni bir remove methodu eklenmesi isteniliyor. Burada LinkedListRec classı kitaptan alınmalıdır. Part 3 de ise 3 adet recursive method bulunmakta. Aslında bu 3 method recursive değil, wrapper görevi görerek recursive olan diğer methodları çağırmaktalar. Bu 3 method için, içinde 2 adet List bulunduran bir class gerekli.

The Project usecase diagrams (extra points)

Step	User's Action	System's Response
1	Call TowerOfHanoi method	Print the solution of TowerOfHanoi method for the given disk size.
2	Call remove method	removes all duplicate elements in linked list.
3	Call intersectionOfLists method	returns intersection set as a list of list1
4	Call unionOfLists method	returns union set as a list of list1 and list 2
5	Call isSubset method	return true if list2 is subset of list1
6	Call HowManyTimesContain method	return *how Many times used element in the given list*

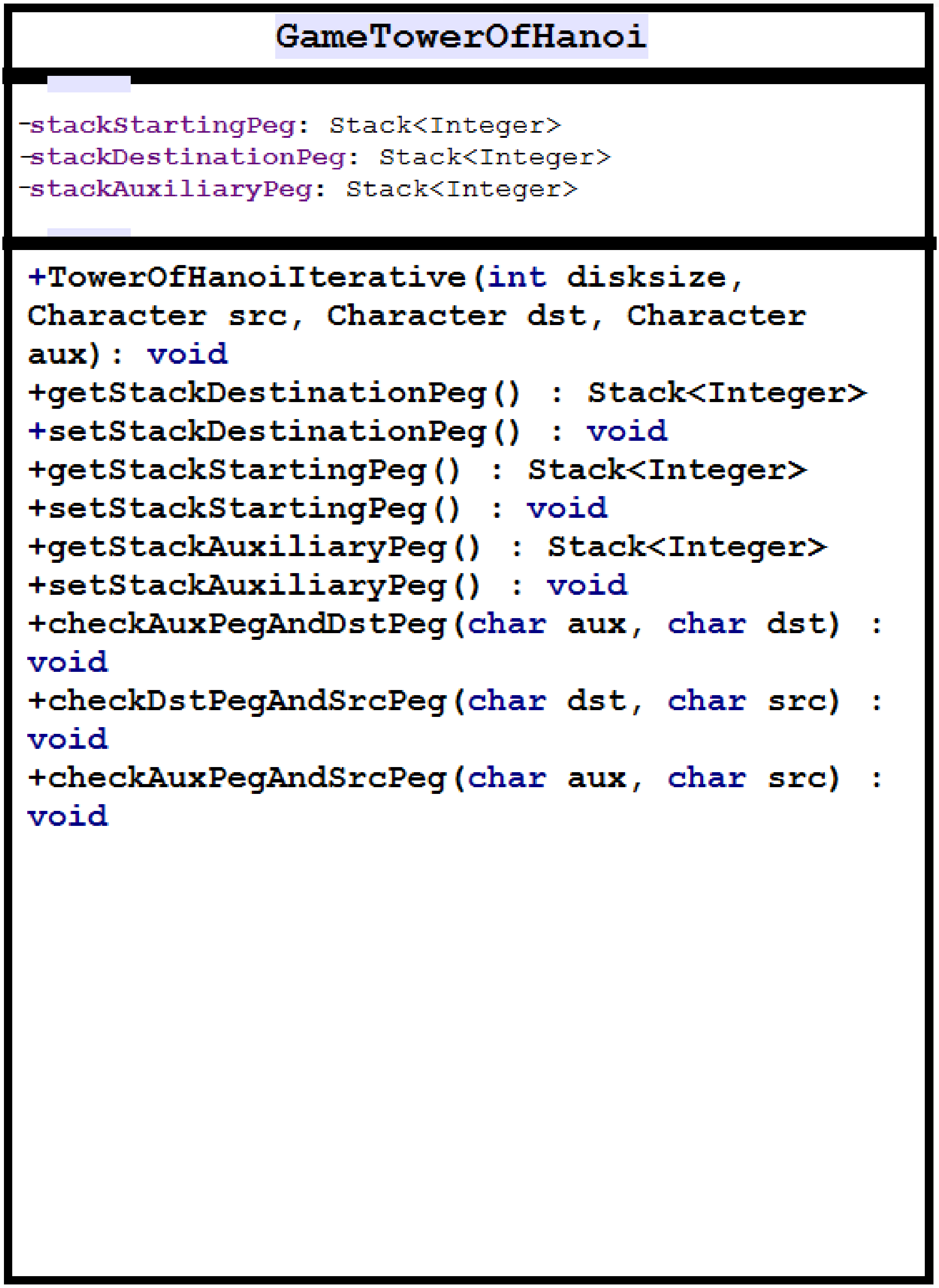
Class Diagrams:



ClassRec

```
- list1 : List<E>

- list2 : List<E>
```

```
+getList1(): List
+setList1(List newList1): void
+setList1(): void
+getList2(): List
+setList2(List newList2): void
+setList2(): void
+sortList(Collection<? extends E>
+collection):List<E>
+list1ToString(): String
+list2ToString(): String
+intersectionOfLists(): List<E>
+intersectionOfLists(int indexOfList1, int
+indexOfList2): List<E>
+unionOfLists(): List<E>
+unionOfLists(int indexOfList1, int
+indexOfList2)
+boolean isSubset(): List<E>
+isSubset: boolean
+howManyContain(List<E> list, E element):
int
```



GameTowerOfHanoi

-stackStartingPeg: Stack<Integer>
-stackDestinationPeg: Stack<Integer>
-stackAuxiliaryPeg: Stack<Integer>

+TowerOfHanoiIterative(int disksize, Character src, Character dst, Character aux): void
+getStackDestinationPeg() : Stack<Integer>
+setStackDestinationPeg() : void
+getStackStartingPeg() : Stack<Integer>
+setStackStartingPeg() : void
+getStackAuxiliaryPeg() : Stack<Integer>
+setStackAuxiliaryPeg() : void
+checkAuxPegAndDstPeg(char aux, char dst) : void
+checkDstPegAndSrcPeg(char dst, char src) : void
+checkAuxPegAndSrcPeg(char aux, char src) : void

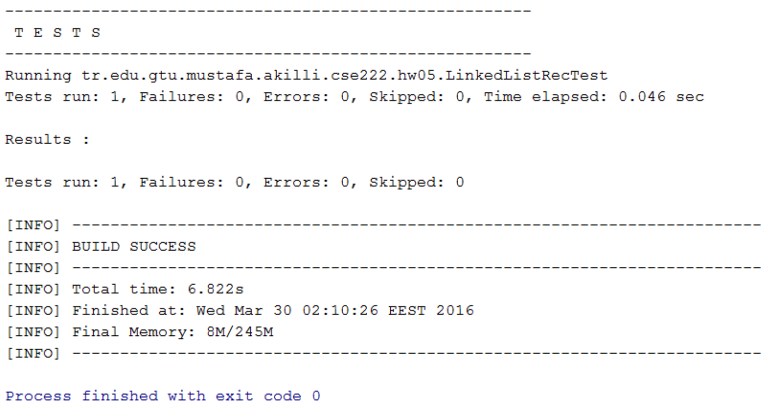
Problem solutions approach

Part 1 için Towers of Hanoi’nin Recursive çözümüne baktım. Çözüm adımlarını inceleyince 3 adımda bir tekrar olduğunu gördüm. Toplam adım sayımız 2^n-1 olmak üzere. İndexin 3’e modunu alarak adımların tekrar tekrar yapılmasını sağladım.

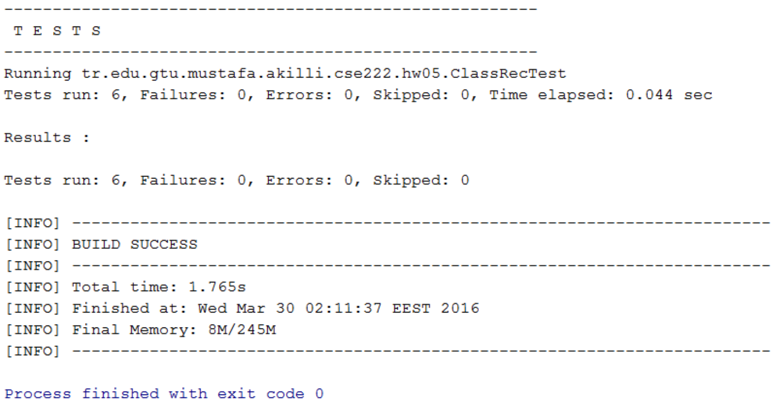
Part 2 için Kitabın kaynak kodlarından LinkedListRec Classını bularak ödevin içine ekledim. Daha sonra parametre alan remove methodunu silerek, bizden istenen remove methodunu classa ekledim. Parametresiz remove methodunuda wrapper olarak kullandı.

Part 3 için bir class oluşturmam gerekiyordu. Methodları recursive olduğundan adını classRec koydum. Daha sonra için iki tane List ekledim ve metodları yazmaya başladım.

Test Cases

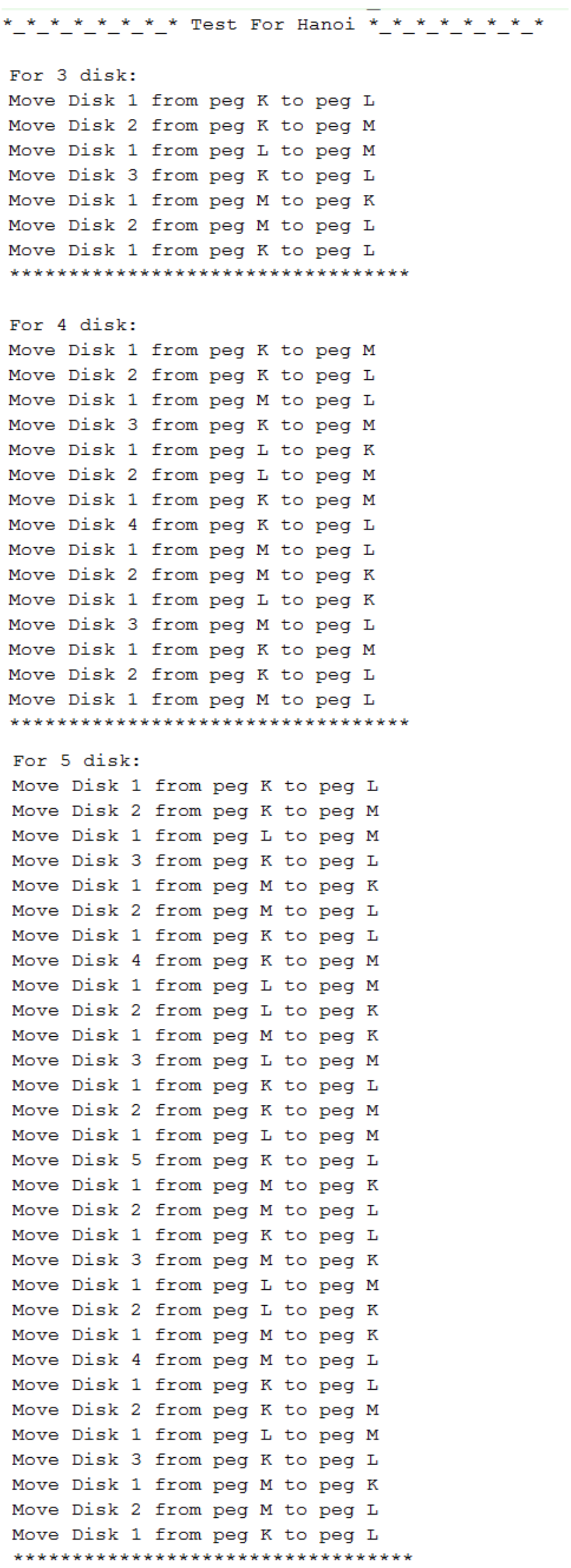


```
-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running tr.edu.gtu.mustafa.akilli.cse222.hw05.LinkedListRecTest
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.046 sec

Results :

Tests run: 1, Failures: 0, Errors: 0, Skipped: 0

[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 6.822s
[INFO] Finished at: Wed Mar 30 02:10:26 EEST 2016
[INFO] Final Memory: 8M/245M
[INFO] ------------------------------------------------------------------------

Process finished with exit code 0
```



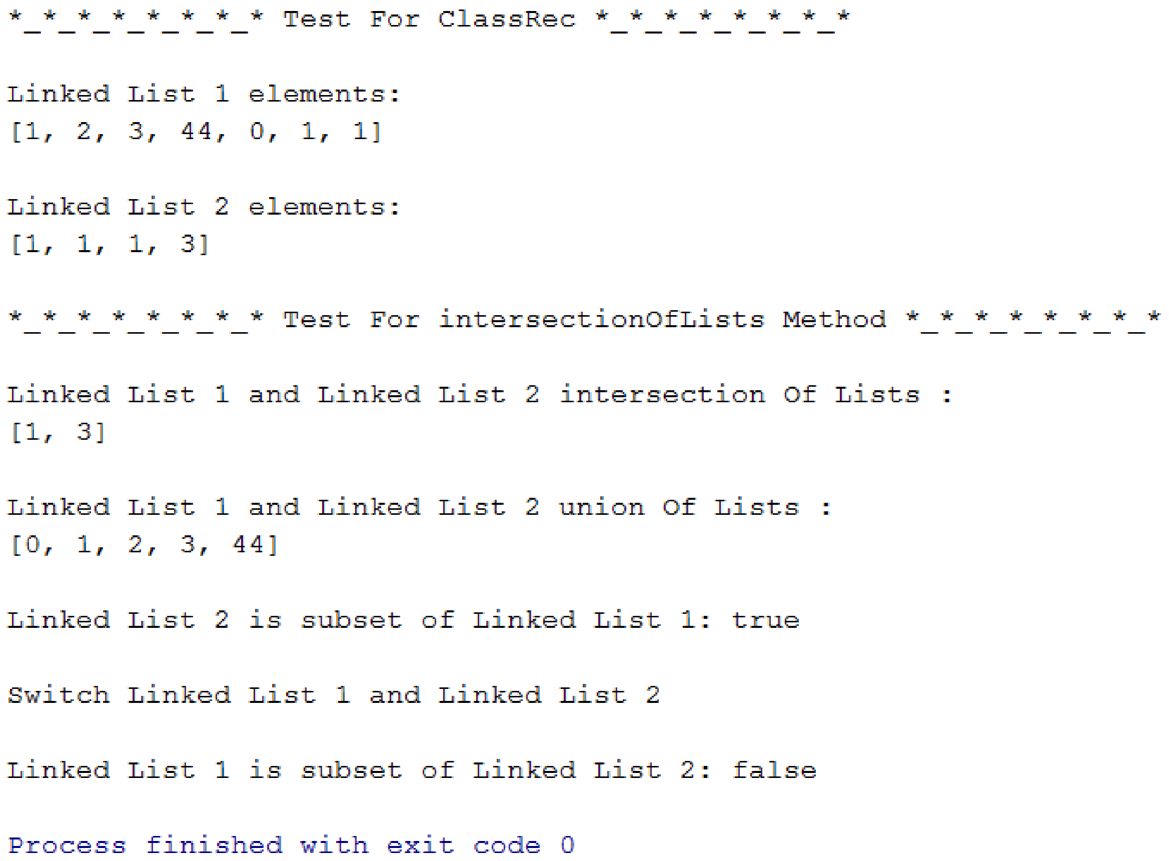
```
-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running tr.edu.gtu.mustafa.akilli.cse222.hw05.ClassRecTest
Tests run: 6, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.044 sec

Results :

Tests run: 6, Failures: 0, Errors: 0, Skipped: 0

[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 1.765s
[INFO] Finished at: Wed Mar 30 02:11:37 EEST 2016
[INFO] Final Memory: 8M/245M
[INFO] ------------------------------------------------------------------------

Process finished with exit code 0
```

Running command and results



```
*_*_*_*_*_*_*_* Test For Hanoi *_*_*_*_*_*_*_*

 For 3 disk:
 Move Disk 1 from peg K to peg L
 Move Disk 2 from peg K to peg M
 Move Disk 1 from peg L to peg M
 Move Disk 3 from peg K to peg L
 Move Disk 1 from peg M to peg K
 Move Disk 2 from peg M to peg L
 Move Disk 1 from peg K to peg L
 ***********************************

 For 4 disk:
 Move Disk 1 from peg K to peg M
 Move Disk 2 from peg K to peg L
 Move Disk 1 from peg M to peg L
 Move Disk 3 from peg K to peg M
 Move Disk 1 from peg L to peg K
 Move Disk 2 from peg L to peg M
 Move Disk 1 from peg K to peg M
 Move Disk 4 from peg K to peg L
 Move Disk 1 from peg M to peg L
 Move Disk 2 from peg M to peg K
 Move Disk 1 from peg L to peg K
 Move Disk 3 from peg M to peg L
 Move Disk 1 from peg K to peg M
 Move Disk 2 from peg K to peg L
 Move Disk 1 from peg M to peg L
 ***********************************

 For 5 disk:
 Move Disk 1 from peg K to peg L
 Move Disk 2 from peg K to peg M
 Move Disk 1 from peg L to peg M
 Move Disk 3 from peg K to peg L
 Move Disk 1 from peg M to peg K
 Move Disk 2 from peg M to peg L
 Move Disk 1 from peg K to peg L
 Move Disk 4 from peg K to peg M
 Move Disk 1 from peg L to peg M
 Move Disk 2 from peg L to peg K
 Move Disk 1 from peg M to peg K
 Move Disk 3 from peg L to peg M
 Move Disk 1 from peg K to peg L
 Move Disk 2 from peg K to peg M
 Move Disk 1 from peg L to peg M
 Move Disk 5 from peg K to peg L
 Move Disk 1 from peg M to peg K
 Move Disk 2 from peg M to peg L
 Move Disk 1 from peg K to peg L
 Move Disk 3 from peg M to peg K
 Move Disk 1 from peg L to peg M
 Move Disk 2 from peg L to peg K
 Move Disk 1 from peg M to peg K
 Move Disk 4 from peg M to peg L
 Move Disk 1 from peg K to peg L
 Move Disk 2 from peg K to peg M
 Move Disk 1 from peg L to peg M
 Move Disk 3 from peg K to peg L
 Move Disk 1 from peg M to peg K
 Move Disk 2 from peg M to peg L
 Move Disk 1 from peg K to peg L
 ***********************************
```

```
*_*_*_*_*_*_*_* Test For Remove Method in ListedListRec *_*_*_*_*_*_*_*


Linked List Rec:

8
8
8
8
8
1
8
4
8
8
8
5
8
8
8
8
3
8
8
8
8


Remove 8, after Linked List Rec:

1
4
5
3

***********************************
```



```
*_*_*_*_*_*_*_* Test For ClassRec *_*_*_*_*_*_*_*

Linked List 1 elements:
[1, 2, 3, 44, 0, 1, 1]

Linked List 2 elements:
[1, 1, 1, 3]

*_*_*_*_*_*_*_* Test For intersectionOfLists Method *_*_*_*_*_*_*_*

Linked List 1 and Linked List 2 intersection Of Lists :
[1, 3]

Linked List 1 and Linked List 2 union Of Lists :
[0, 1, 2, 3, 44]

Linked List 2 is subset of Linked List 1: true

Switch Linked List 1 and Linked List 2

Linked List 1 is subset of Linked List 2: false

Process finished with exit code 0
```